APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/023,117	12/17/2001	Bernardo De Oliveira Kastrup Pereira	NL 000721	2411

24737   7590   11/14/2007
PHILIPS INTELLECTUAL PROPERTY & STANDARDS
P.O. BOX 3001
BRIARCLIFF MANOR, NY 10510

EXAMINER
ELLIS, RICHARD L

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
11/14/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Application Number: 10/023,117
Filing Date: December 17, 2001
Appellant(s): Pereira et al.

MAILED

NOV 1 3 2007

Technology Center 2100

Robert M. McDermott
<u>For Appellant</u>

EXAMINER'S ANSWER

This is in response to the appeal brief filed September 5, 2007 appealing from the Office action mailed June 7, 2007.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

The following is a listing of the evidence (e.g., patents, publications, Official Notice, and admitted prior art) relied upon in the rejection of claims under appeal.

6,006,321   Abbott 12-21-1999

Computer Organization, second edition   V. Carl Hamacher et al.   1984

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1-20 are rejected under 35 USC § 102(b) as being clearly anticipated by Abbott, U.S. Patent 6,006,321. Pages 19-22 of *Computer Organization, second edition* by V. Carl Hamacher et al. (1984) are further cited as a showing of what was known as inherent in the CPU instruction processing art.

Claim Construction

All of applicant's independent claims (1, 5, 6, 7, and 13) utilize the transitional term "comprising" to introduce the body of the claim. As per MPEP 2111.03, the term "comprising" is taken to mean:

> The transitional term "comprising", which is synonymous with "including," "containing," or "characterized by," is inclusive or open-ended and does not exclude additional, unrecited elements or method steps. See, e.g., > *Mars Inc. v. H.J. Heinz Co.*, 377 F.3d 1369, 1376, 71 USPQ2d 1837, 1843 (Fed. Cir. 2004) ("like the term comprising,' the terms containing' and mixture' are open-ended.").< *Invitrogen Corp. v. Biocrest Mfg., L.P.*, 327 F.3d 1364, 1368, 66 USPQ2d 1631, 1634 (Fed. Cir. 2003) ("The transition comprising' in a method claim indicates that the claim is open-ended and allows for additional steps."); *Genentech, Inc. v. Chiron Corp.*, 112 F.3d 495, 501, 42 USPQ2d 1608, 1613 (Fed. Cir. 1997) ("Comprising" is a term of art used in claim language which means that the named elements are essential, but other elements may be added and still form a construct within the scope of the claim.); *Moleculon Research Corp. v. CBS, Inc.*, 793 F.2d 1261, 229 USPQ 805 (Fed. Cir. 1986); *In re Baxter*, 656 F.2d 679, 686, 210 USPQ 795, 803 (CCPA 1981); *Ex parte Davis*, 80 USPQ 448, 450 (Bd. App. 1948) ("comprising" leaves "the claim open for the inclusion of unspecified ingredients even in major amounts"). >In *Gillette Co. v. Energizer Holdings Inc.*, 405 F.3d 1367, 1371-73, 74 USPQ2d 1586, 1589-91 (Fed. Cir. 2005), the court held that a claim to "a safety razor blade unit comprising a guard, a cap, and a group of first, second, and third blades" encompasses razors with more than three blades because the transitional phrase "comprising" in the preamble and the phrase "group of" are presumptively open- ended. "The word comprising' transitioning from the preamble to the body signals that the entire claim is presumptively open-ended." Id.

In the claims in this case, applicant has utilized the term "comprising" to transition from the preambles to the bodies of each claim. Therefore as detailed in MPEP 2111.03, this "signals that the entire claim is presumptively open-ended". As seen from MPEP 2111.03, an open-ended claim "is inclusive ... and does not exclude additional, unrecited elements or method steps." It also means that "other elements may be added and still form a construct within the scope of the claim." And much like the *Gillette Co. v. Energizer Holdings Inc.* decision, applicant's open-ended claims encompass data processing systems with more elements (in any amount) than that which is recited in the claim language itself.

"Instruction" vs. "configurable function" vs. "input ordering instruction",

"configured logic function", and "output ordering instruction"

Taking claim 1 as exemplary, the claim introduces "an instruction" at line 2. The introduction of "an instruction" is that the instruction is part of a program that the data processing device is capable of executing. Crossing lines 3-4 the claim introduces "a configurable function" which is utilized by a "configurable function unit" for executing the instruction which was introduced at line 2. Because the claim has introduced a new element ("a configurable function") and defined this new element as being used in the execution of the instruction introduced on line 2, this new element ("a configurable function") is interpreted as separate from the instruction of line 2.

At lines 4-6 the claim defines what is "included" in "the configured function" introduced on lines 3-4. The claim language defines "the configured function" as including "an input ordering instruction", "a configured logic function", and "an output ordering instruction". Therefore, despite the fact that the claim language has now reused the word "instruction" for two elements that are included within "the configured function", the two "instructions" ("input ordering instruction" and "output ordering instruction") are separate and distinct elements from "the instruction" introduced on line 2. This is because the claim defined "the configured function" as a separate element apart from "the instruction" of line 2, accordingly, sub-elements of "the configured function" must therefore be separate from "the instruction" of line two.

Accordingly, the claim language contains three similarly named, but separate, elements ("an instruction", "an input ordering instruction", and "an output ordering instruction") where the second two elements (the "input ordering" and "output ordering" instructions) are not parts or sub-elements of the first element (the "instruction").

Applied Art operation

The following is a short summary of Abbott's description of how the invention of the applied patent operates. It is placed here because it references several different citations of Abbott in sequence and would become confusing if interlaced into the rejection writeup below.

Abbott's system operates under control of what Abbott refers to as a "Direct Control Vector" ("DCV"). Abbott defines the meaning of a DCV at col. 1 lines 21-25:

"The control signals that control the operations of the datapath may be considered as a vector of

bits, which is known as a "direct control vector", since it directly controls the datapath operations."

Accordingly, Abbott utilizes DCV to mean the bits that "directly control the datapath operations".

Abbott additionally defines another control vector, termed the "indirect control vector" ("ICV") at col. 1 lines 29-36:

"Typically, the direct control vector is developed from a combination of bits in the instruction, processor state bits (which are sometimes known as "mode bits"), and logic gates. The combination of instruction bits and mode bits, all of which may change on each cycle, can be considered as an "indirect control vector" since it indirectly controls the datapath operations."

Accordingly, when Abbott refers to an ICV, he is referring to the combination of instruction and mode bits that are used to develop (or create) a particular DCV.

Abbott gives a short example of this operation at col. 1 lines 38-44:

"For example, when an ADD instruction is issued in a CPU, an opcode (the indirect control vector) that is contained in the ADD instruction is decoded by the control mechanism to generate appropriate control signals (the direct control vector) to cause the ALU to add the two operands indicated by the ADD instruction"

At col. 3 lines 26-31 Abbott details how in the invention, an ICV is decoded into a DCV:

"In one embodiment, the field programmable device includes circuitry for decoding indirect control vectors into direct control vectors that specify the operation(s) to be performed by the programmable logic datapath on a cycle by cycle basis."

At col. 6 lines 7-12 Abbott details how the ICV is obtained and/or derived from an instruction:

"In an alternative embodiment, the control logic unit 110 is not present and the decoding logic unit 112 is operatively coupled to receive the indirect control vectors from the input array 102 (e.g., an opcode contained in an instruction provided by a microprocessor that may be coupled to the field programmable device 100)."

At col. 5 lines 46-60 Abbott explains that the DCV is used to control/operate all the aspects of the device:

"In one embodiment, the decoding logic unit stores 16 DCVs for the subset selection portion (described later herein) of the programmable logic datapath   114 in a random access memory (RAM) and 16 DCVs in a ROM. The memory width needed to control this portion of the programmable logic datapath in this embodiment is 1536 bits, while the length of the indirect ICV sub-field that addresses this memory is 5 bits. Other memories control other portions of the selected DCV, applying them as control signals to the selector unit 108, the register bank 106, the register bank 104, other parts of the programmable logic datapath 114, and/or the programmable arithmetic datapath 116. In one embodiment, the total DCV length when all the fields are accounted for is 2669 bits."

Accordingly, in the rejections below, when applicant claims "an instruction" this is equivalent to Abbott's disclosed "instruction". However, when applicant claims "an input ordering instruction", "a configured logic function", and "a output ordering instruction", the functionality and operation of these claim elements are provided by Abbott's "direct control vector" ("DCV"). This is because as claimed, applicant's "input ordering", "configured logic", and "output ordering" instructions/function are claimed as controlling/configuring the operation of applicant's device, in the exact same manner that Abbott's DCV is disclosed as controlling/configuring the device of the prior art. In this manner, Abbott's DCV is equivalent to applicant's claimed "configured function" in that Abbott's DCV contains the claimed "input ordering instruction", "configured logic function", and "output ordering instruction".

Abbott taught (e.g. See figs 1-7b) the invention as claimed (as per claim 1), including a data processing ("DP") system comprising:

1. A data processing device	Fig. 7a, showing a programmable system chip containing memory, cpu, i/o and field programmable devices, which together make a "data processing device".
configured according to a device configuration	Col. 5 lines 20-35, specifically lines 32-35 detailing that ICV selects DCV to "program/configure various portions of the field programmable device 100", the DCV producing within the device a "device configuration" (col. 5 lines 54-58).
so as to be capable of executing a program comprising an instruction, the device comprising	Col. 6 lines 10-11, detailing that input array 102 receives an instruction which directs operation of the device (note that as detailed above, this "instruction" is separate from the below recited "input ordering" and "output ordering" instructions).
a configurable functional unit	Fig. 1, note specifically that "PROGRAMMABLE LOGIC DATAPATH" is a "configurable" functional unit in that by being "programmable", it is "configurable". Note also that Abbott refers to the device as configurable at least at col. 5 lines 31-35.
for executing the instruction according to a configurable function	Col. 3 lines 14-23, detailing that the function performed is programmable (configurable) on a cycle-by-cycle basis. Col. 5 lines 30-35 and 46-60 detailing that the DCV configures and controls the Abbott's reconfigurable system to implement a function, and is therefore "a configurable function" as per the claim language.

that is configured outside the instruction,	Col. 5 lines 47-60, which detail that the control and configuration of the device is as a result of the DCV, the DCV being derived from, but not part of, the input instruction, col. 5 lines 20-35 detailing that DCV is derived from ICV and col. 6 lines 7-12 detailing that ICV is derived from an instruction. As a result, the "configuration" which happens via the DCV is "outside" of the instruction because it is only indirectly derived from the instruction by two levels of indirection (instruction -> ICV -> DCV).
the configured function including an input ordering instruction,	Fig. 2, 202, fig. 2 shows details of the "PROGRAMMABLE LOGIC DATAPATH" of fig. 1, unit 202 of fig. 2 is a "REARRANGEMENT CIRCUIT" which reorders input data bits (col. 6 line 37 to col. 7 line 25), note specifically col. 6 lines 54-58 detailing that unit 202 allows for "selectively routing any of the 48 input bits to its output" and lines 61-63 detailing "Thus, the 48 multiplexing circuits operate to dynamically select and/or rearrange (i.e., modify relative bit positions) the input bits" this rearrangement being an "ordering function". Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "REARRANGEMENT CIRCUIT" and therefore, the DCV is providing "an input ordering instruction" because it "instructs" the "REARRANGEMENT CIRUCIT" as to how to rearrange (reorder) the bits.
a configured logic function, and	Col. 3 lines 6-22, note specifically lines 19-22 which state "the programmable logic datapath of the invention can provide dynamic programmability on a cycle-by-cycle basis to perform a number of **logic operations** on inputs of various lengths and outputs" where a "logic operation" is the claimed "logic function".
an output ordering instruction,	Fig. 4, 410 "TRANSPOSITION CIRCUIT", fig. 4 is an internal view of element 400 of fig. 2, the operation of which is detailed at col. 12 line 1 to col 13 line 43. Note specifically col. 12 lines 8-10 which state "The set of multiplexers provide optional transposition (i.e., positional interchange) of rotate bit groups" where "positional interchange" is a change in the ordering of bits and is an ordering operation. Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "TRANSPOSITION CIRCUIT" and therefore, the DCV is providing "an output ordering instruction" because it "instructs" the "TRANSPOSITION CIRCUIT" as to how to transpose (rearrange, reorder) the bits.
the configurable functional unit including	Fig. 2.

a unit input	Fig. 1, 108, showing SELECTOR UNIT 108 as providing input data to PROGRAMMABLE LOGIC DATAPATH 114. Fig. 2, showing a 48 bit wide "INPUT" on the left hand side of the figure to unit 114 (which fig. 2 shows internal details of unit 114 of fig. 1.
for inputting a plurality of input bits	Input bits flow across the line from unit 108 to unit 114, note further fig. 2 showing a 48 bit wide "INPUT" at the left hand side, 48 bits being a "plurality of input bits".
of one or more source registers specified by the instruction,	Fig. 1, showing two register banks at 104 and 106 providing data to selector unit 108 to form input for unit 114. Note that the cited Hamacher et al. reference supports that instructions inherently specify one or more source registers. Additionally, Abbott indicates at col. 1 lines 38-44 that he was aware of the inherent fact that instructions specify source registers when he states "to cause the ALU to add the <u>two operands</u> indicated by the ADD instruction".
a unit output	Fig. 1, showing a line exiting unit 114, note that fig. 2 further shows a 64 bit wide "OUTPUT" from unit 114.
for outputting a plurality of output bits	Fig. 2, showing a 64 bit "OUTPUT" on the right hand side, 64 bits being a "plurality of output bits".
to a destination register specified by the instruction,	Fig. 1, showing line 120 looping from output of unit 114 to register banks 104 and 106, thereby providing a result to a destination register specified by the instruction (col. 19 lines 63-66). Note that Hamacher et al. details that all instructions specify source and destination registers, the number directly specified varies, and the manner with which the instruction specifies the source and destination registers varies, but all instructions specify source and destination registers. Note on pg. 19, the Add A,B,C instruction directly specifies three registers, two sources (A,B) and a destination (C). Furthermore, Abbott taught that the patented invention processed instructions (col. 6 lines 7-12), and indicates at col. 1 lines 38-44 that he was aware of the inherent fact that instructions specify source registers when he states "to cause the ALU to add the two operands indicated by the ADD instruction".

a first programmable connection circuit	Fig. 2, 202, fig. 2 shows details of the "PROGRAMMABLE LOGIC DATAPATH" of fig. 1, unit 202 of fig. 2 is a "REARRANGEMENT CIRCUIT" which reorders input data bits (col. 6 line 37 to col. 7 line 25), note specifically col. 6 lines 54-58 detailing that unit 202 allows for "selectively routing any of the 48 input bits to its output" and lines 61-63 detailing "Thus, the 48 multiplexing circuits operate to dynamically select and/or rearrange (i.e., modify relative bit positions) the input bits". This change in relative position of bits being a change in "connection" of bits from the input to the output of the circuit (i.e., a bit input at position 2 is connected to an output at position 12), thereby disclosing a "programmable connection circuit" as claimed.
that is configured to receive the plurality of input bits and	Fig. 2, showing the REARRANGEMENT CIRCUIT receiving the 48 INPUT bits on the left hand side of the figure.
selectively route the input bits	Col. 6 lines **61-63** detailing **"Thus, the 48 multiplexing circuits operate to dynamically select and/or rearrange (i.e., modify relative bit positions) the input bits".** The modification of relative bit position being the claimed "selectively rout[ing] the input bits" because, e.g., a bit input at position 2 is routed to an output at position 12.
to provide a set of logic input bits, based on the input ordering instruction,	Fig. 2, showing that REARRANGEMENT CIRCUIT 202 provides a set of "logic input bits" (48 bits shown connecting from unit 202 into logic 210), these 48 bits being "input" to the next logic unit. Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "REARRANGEMENT CIRCUIT" and therefore, the DCV is providing "an input ordering instruction" because it "instructs" the "REARRANGEMENT CIRUCIT" as to how to rearrange (reorder) the bits.
a plurality of independent configurable logic blocks	Fig. 7a, 702a ... 702; fig. 2, 210, col. 7 line 25 to col. 8 line 45, note specifically col. 7 lines 29-32 which details "The selective field negation circuit 210 selectively negates certain bits output by the rearrangement circuit 202 based on control bits provided by the decoding logic unit 112", each sub element shown inside element 210 on fig. 2 being a "logic block" that is "independently configurable" (because if each block were not independently configurable, the defined "selective negation" would not occur).
for performing programmable logic operations	Col. 7 line 25 to col. 8 line 45, selective negation being a "programmable logic operation".
to produce a set of logic output bits corresponding to the configured logic function being applied to the set of logic input bits,	Fig. 2 shows unit 210 outputting 48 bits of output to unit 212, this output being the result of the "configured logic function" of unit 210, that function having been applied to the 48 input bits provided to unit 210 from unit 202.

a second programmable connection circuit,	Fig. 4, 410, note that fig. 4 is an internal detail diagram of unit 212 from fig. 2. Note that unit 410 is titled "transposition circuit" where "transposition" means change the order (or connection) through the unit (col. 12 lines 8-11).
that is configured to receive the set of logic output bits	Fig. 4, unit 410 receives the 48 bits provided as output from unit 210 as shown on the left hand side of fig. 4, unit 410 simply receives these bits by way of unit 402, 404, 406, and 408. However, as detailed in the claim construction section above, applicants claims are written using inclusive or open-ended language and as such, the claims do not exclude there being additional elements present between the claimed "configurable logic blocks" and claimed "second programmable connection circuit". The open-ended language allows for "other elements [to] be added and still form a construct within the scope of the claim." (*Genentech, Inc. v. Chiron Corp.*) The "other elements" being elements 402, 404, 406 and 408 which in the prior art form a conduit for the bits from the programmable logic circuit to the second programmable connection circuit.
and selectively route the logic output bits	Col. 12 lines 5-14, specifically lines 8-11 which state "The set of multiplexers provide optional transposition (i.e., positional interchange) of rotate bit groups between the set of reduction networks in the reduction network bank 212." where "positional interchange" is "selectively rout[ing]" as claimed.
to provide the plurality of output bits, based on the output ordering instruction.	Fig. 4 shows that unit 410 provides output bits (arrow connecting 410 to 412) that flow through a conduit formed of unit 412, 414, optional functional unit 214 of fig. 2, to the OUTPUT shown on the right side of fig. 2, connecting to line 120 of fig. 1, which itself connects into register banks 104 and 106. Therefore, bits shown leaving unit 114 on line 120 of fig. 1 are "provided" by unit 410 by flowing through units 412, 414, and optional unit 214. As detailed above, applicant's use of the open-ended term "comprising" allows for "other elements [to] be added and still form a construct within the scope of the claim" (*Genentech, Inc. v. Chiron Corp.*). The "other added" elements are elements 412, 414, and optional unit 214.

Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "TRANSPOSITION CIRCUIT" and therefore, the DCV is providing "an output ordering instruction" because it "instructs" the "TRANSPOSITION CIRCUIT" as to how to transpose (rearrange, reorder) the bits. |

As to claims 2 and 3, Abbott taught each logic block having a plurality of outputs (fig. 2, 48 bits from 202, 48 bits from "SELECTIVE FIELD NEGATION CIRCUIT", 64 bits from "REDUCTION

NETWORK BANK", and 64 bits from 214), at least one of the bits of the unit output being connectible exclusively to one of the outputs of each logic block (col. 12 lines 8-10 and 24-39), the second programmable connection circuit comprising a multiplexer (col. 12 lines 5-11) for coupling the one of the outputs of a selected one of the logic blocks to the at least one of the bits of the unit output.

As to claim 4, Abbott taught that either the first programmable connection circuit or the second programmable connection circuit having a fixed, unprogrammable connection to an input or output of one of the independent configurable logic blocks and a programmable connection to a remainder of the inputs and outputs (col. 6 line 64 to col. 7 line 25, wherein grouping bits into sets provides a "fixed unprogrammable connection" of that set of bits, i.e., the bits within the set can not each be independently connected).

5. A method of programming	Col. 3 lines 5-8.
a configurable processing device	Fig. 1, col. 4 lines 48-55, note that Abbott defines "field programmable" at col. 2 lines 24-36.
according to a device configuration	Col. 5, lines 31-35 and 46-60, the DCV being the claimed "device configuration" because it "configures" the "device"
to perform a processing task,	Col. 3 lines 16-23, col. 16 line 64 et seq.
wherein the device includes a configurable processing unit	Fig. 1, 114, 116.
that includes one or more programmable logic blocks,	Fig. 1, 114, fig. 2, 202, 210, 212, 214, fig. 4, 402,4 04, 406, 408, 410, 412, 414.
the method comprising:	
identifying a special complex of operations that occurs in the task and requires one or more operand data words and produces a result data work;	Col. 16 line 64 to col. 18 line 30.
searching for an assignment of logic operations for producing different bits of the result data word to different ones of the programmable logic blocks,	Abbott details at col. 5 lines 35-46 that a DCV is created from contents of multiple different memories wherein different portions of the ICV are utilized to locate the appropriate DCV portion from each memory (col. 5 lines 37-41). This is a "searching" (to find/locate) function in that each ICV subfield is used to "find" or "locate" the proper DCV value from each of the multiple memories. The DCV, as detailed previously, being utilized to determine logic operations and bit routing within Abbott's system.

so that the logic operations for producing a subset of the bits of the result data word that, if implemented together in one of the programmable logic blocks, would exceed the capacity of that one of the programmable logic blocks, are distributed over different ones of the logic blocks;	Figs. 7a and 7b detail plural FPD devices in Abbott's embodiment. It would have been inherent that in order to fully utilize the plurality of FPD devices that operates must have been distributed across the plurality of logic blocks in an appropriate manner. To do otherwise would mean that much of Abbott's invention would have not been utilized.
programming each of the programmable logic blocks to perform the logic operations for the bits of the result data word assigned to it;	Col. 5 lines 30-35 and 46-60 detailing that the DCV configures (or programs) the logic blocks to perform a desired operation.
programming a first connection circuit to the programmable logic blocks so as to perform a first routing of bits of an operand of a special instruction to the programmable logic blocks that use those bits of the operand in the logic operations;	Fig. 2, 202, fig. 2 shows details of the "PROGRAMMABLE LOGIC DATAPATH" of fig. 1, unit 202 of fig. 2 is a "REARRANGEMENT CIRCUIT" which reorders input data bits (col. 6 line 37 to col. 7 line 25), note specifically col. 6 lines 54-58 detailing that unit 202 allows for "selectively routing any of the 48 input bits to its output" and lines 61-63 detailing "Thus, the 48 multiplexing circuits operate to dynamically select and/or rearrange (i.e., modify relative bit positions) the input bits" which is an "ordering function". Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "REARRANGEMENT CIRCUIT" and therefore, the DCV is providing "an input ordering instruction" because it "instructs" the "REARRANGEMENT CIRUCIT" as to how to rearrange (reorder) the bits. Note that at least block 210 of fig. 2 shows "programmable logic blocks" contained within the "SELECTIVE FIELD NEGATION CIRCUIT". The input of block 210 is provided by the output from block 202 as shown on fig. 2

and programming a second connection circuit subsequent to the logic blocks so as to perform a second routing of outputs of the programmable logic blocks to bits of the result data word to which the programmable logic blocks are assigned.	Fig. 4, 410 "TRANSPOSITION CIRCUIT", fig. 4 is an internal view of element 400 of fig. 2, the operation of which is detailed at col. 12 line 1 to col 13 line 43. Note specifically col. 12 lines 8-10 which state "The set of multiplexers provide optional transposition (i.e., positional interchange) of rotate bit groups" where "positional interchange" is a change in the ordering of bits and is an ordering instruction. Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "TRANSPOSITION CIRCUIT" and therefore, the DCV is providing "an output ordering instruction" because it "instructs" the "TRANSPOSITION CIRCUIT" as to how to transpose (rearrange, reorder) the bits. Note that block 401, being contained in block 212 of fig. 2, is subsequent to block 210 because it comes after block 210 in the diagram of fig. 2.

6. A method of executing a program	Col. 16 line 64 et seq. giving several different examples of programs executed by Abbott's system
with a processing device	Fig. 7a, showing a processing device
with a configurable functional unit	Fig. 1, showing a "FIELD PROGRAMMABLE DEVICE", note that Abbott defines "field programmable" at col. 2 lines 24-30.
according to a device configuration,	Col. 5 lines 30-35 and 46-60 wherein the DCV provides a "device configuration" by "configuring" the "device".
the method comprising:	
inputting one or more words of one or more operands of a program instruction into the configurable functional unit,	Col. 6 lines 7-12 detailing that an instruction is input to derive the ICV, and as detailed above, the DCV is further derived from the ICV. Note that in order to derive the DCV as detailed by Abbott, an "instruction" must have been input. As well, note that the Hamacher et al. reference supports the fact that all instructions inherently specify one or more operands. Abbott also recognizes this fact at col. 1 lines 39-44 when he states: "to cause the ALU to add the two **operands** indicated by the ADD instruction".
each word including a plurality of bits;	Fig. 2, showing 48 bit of input "word" to the programmable device.

selectively coupling the bits of the words of the operands to inputs of logic blocks, dependent on a configured function of the configurable functional unit;	Fig. 2, 202, fig. 2 shows details of the "PROGRAMMABLE LOGIC DATAPATH" of fig. 1, unit 202 of fig. 2 is a "REARRANGEMENT CIRCUIT" which reorders input data bits (col. 6 line 37 to col. 7 line 25), note specifically col. 6 lines 54-58 detailing that unit 202 allows for "selectively routing any of the 48 input bits to its output" and lines 61-63 detailing "Thus, the 48 multiplexing circuits operate to dynamically select and/or rearrange (i.e., modify relative bit positions) the input bits" which is an "ordering function".
	Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "REARRANGEMENT CIRCUIT" and therefore, the DCV is providing "an input ordering instruction" because it "instructs" the "REARRANGEMENT CIRUCIT" as to how to rearrange (reorder) the bits.
performing programmable logic operations to implement the configured function to provide an output word that includes a plurality of bits;	Col. 5 lines 30-35 and 46-60 detailing that the DCV configures the programmable datapath.
	Col. 6 line 27 to col. 15 line 5 providing detailed information about the logic operations the datapath was programmable to perform. Note specifically, for example, col. 7 line 25 to col 8 line 10 detailing selective negation of bits, selective negation being a "logic operation".
selectively coupling bits of the output word to bits of a result word, dependent on the configured function; and	Fig. 4, 410 "TRANSPOSITION CIRCUIT", fig. 4 is an internal view of element 400 of fig. 2, the operation of which is detailed at col. 12 line 1 to col 13 line 43. Note specifically col. 12 lines 8-10 which state "The set of multiplexers provide optional transposition (i.e., positional interchange) of rotate bit groups" where "positional interchange" is a change in the ordering of bits and is an ordering instruction.
	Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "TRANSPOSITION CIRCUIT" and therefore, the DCV is providing "an output ordering instruction" because it "instructs" the "TRANSPOSITION CIRCUIT" as to how to transpose (rearrange, reorder) the bits.
	Note that the output of block 410 on fig. 4 is carried by the conduit formed by blocks 412 and 414 to the output of block 410 (right hand side of fig. 4) which is subsequently carried via optional unit 214 on fig. 2 to become the output of block 114 on fig. 1, thereby becoming the "result" (or output) of the programmable logic circuit detailed by Abbott.

outputting the result word to a destination identified in the program instruction.	Fig. 2, right hand side of figure showing "OUTPUT" of 64 bits of result word, which is then shown on fig. 1 as line 120 connecting to register banks 104 and 106 for storage as determined by the instruction which derived the ICV which in turn derived the DCV which controlled the device.

7. A data processing device comprising:	Fig. 7a, showing a "data processing device"
a register file	Fig. 1, 104, 106, showing "REGISTER BANK"s
that includes a plurality of registers,	Col. 6, lines 59-61, detailing that the "REGISTER BANK"s contain plural registers.
each register including a word,	Col. 6, lines 66-67, detailing that the registers within the banks are 32 bits wide, 32 bits equaling "a word".
each word including a plurality of bits in a first bit order, and	Col. 6, lines 66-67, detailing that "a word" as explained immediately above comprises 32 bits, the bits will inherently have "a first bit order" (i.e., arrangement of bits).
at least one configurable function units	Fig. 1, showing a "FIELD PROGRAMMABLE DEVICE", note that Abbott defines "field programmable" at col. 2 lines 24-30.
that is dynamically configurable	Col. 3 lines 16-22, detailing that the system "can provide dynamic programmability" which is "dynamically configurable"
to effect different functions	Col. 3 lines 16-22, detailing "a number of logic operations" which are "different functions"
at different times,	Col. 3 lines 16-22, detailing programmability on a "cycle-by-cycle basis" which is "different times".
based on received configuration data,	Col. 5 lines 29-35 and 46-60 detailing that the DCV configures and controls the device and therefore the DCV is "configuration data".
the at least one configurable function unit including:	

a first connection circuit that is configured to receive one or more words from the plurality of registers, and to provide one or more words of bits in a second bit order based on the received configuration data,	Fig. 2, 202, fig. 2 shows details of the "PROGRAMMABLE LOGIC DATAPATH" of fig. 1, unit 202 of fig. 2 is a "REARRANGEMENT CIRCUIT" which reorders input data bits (col. 6 line 37 to col. 7 line 25), note specifically col. 6 lines 54-58 detailing that unit 202 allows for "selectively routing any of the 48 input bits to its output" and lines 61-63 detailing "Thus, the 48 multiplexing circuits operate to dynamically select and/or rearrange (i.e., modify relative bit positions) the input bits" which is an "ordering function".

Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "REARRANGEMENT CIRCUIT" and therefore, the DCV is providing "an input ordering instruction" because it "instructs" the "REARRANGEMENT CIRUCIT" as to how to rearrange (reorder) the bits.

The connection circuit 202 is shown receiving 48 bits of "INPUT" on the left hand side of fig. 2, that input shown arriving from "REGISTER BANK" 104 or 106 via "SELECTOR UNIT". |
| one or more programmable logic blocks that are configured to receive the one or more words in the second bit order from the first connection circuit, and to provide an output word of bits in a first output bit order based on the received configuration data, | Fig. 2, "SELECTIVE FIELD NEGATION CIRCUIT" 210, shown receiving 48 bits of data from "REARRANGEMENT CIRCUIT" 202. The action of "selective field negation" being a "programmable logic" and box 210 showing plural "blocks" within box 210 on fig. 2.

Box 210 is also shown providing an output 48 bits, this output set of bits will inherently be in a "first output bit order", i.e., the order with which the logic block outputs inside block 210 are connected to the 48 lines exiting block 210.

Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "SELECTIVE FIELD NEGATION CIRCUIT". |

a second connection circuit that is configured to receive the output word and to provide therefrom a word of bits in a second output bit order that is stored in a destination register of the plurality of registers based on the received configuration data.	Fig. 4, 410 "TRANSPOSITION CIRCUIT", fig. 4 is an internal view of element 400 of fig. 2, the operation of which is detailed at col. 12 line 1 to col 13 line 43. Note specifically col. 12 lines 8-10 which state "The set of multiplexers provide optional transposition (i.e., positional interchange) of rotate bit groups" where "positional interchange" is a change in the ordering of bits and is an ordering instruction. Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "TRANSPOSITION CIRCUIT" and therefore, the DCV is providing "an output ordering instruction" because it "instructs" the "TRANSPOSITION CIRCUIT" as to how to transpose (rearrange, reorder) the bits. Note, as detailed in the claim construction section above, applicants claims are written using inclusive or open-ended language and as such, the claims do not exclude there being additional elements present between the claimed "programmable logic blocks" and claimed "second connection circuit". The open-ended language allows for "other elements [to] be added and still form a construct within the scope of the claim." (Genentech, Inc. v. Chiron Corp.) The "other elements" being elements 402, 404, 406 and 408 which in the prior art form a conduit for the bits from the programmable logic blocks to the second connection circuit. Note that fig. 4 details that the output from element 400 is input to optional element 214 of fig. 2, fig. 2 shows that the output of optional element 214 is the "OUTPUT" of block 114, and fig. 1 shows that the output of block 114 is carried over lines 120 to register banks 104 and 106. Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "TRANSPOSITION CIRCUIT".

As to claim 8, Abbott taught an input unit (fig. 1, "SELECTOR UNIT" 108) that was configured (connection between "DECODING LOGIC UNIT" 112 and unit 108) to receive the one or more words from the plurality of registers (104, 106) and to provide the one or more words to the first connection circuit (unit 108 connects to datapath 114, fig. 2 shows datapath 114 receiving as INPUT to first connection circuit (202) the bits from unit 108 on fig. 1).

As to claims 9 and 10, Abbott taught an output unit (fig. 2, 214) that was configured (line from

112 to 214) to receive the one or more words from the second connection circuit (fig. 2 sows unit 214 receiving 64 bits output from unit 212, unit 212 containing the second connection circuit), and to provide the one or more words to the destination register (fig. 2 shows unit 214 providing 64 bits of "OUTPUT" which fig. 1 shows connecting via line 120 to register banks 104, 106).

As to claim 11, Abbott taught that source registers of the one or more words from the plurality of registers were identified in an instruction that effects execution of a current function of the at least one configurable function unit (col. 6 lines 7-12, col. 1 lines 38-44, furthermore, it is inherent that instructions specify operands in modern computing systems).

As to claim 12, although Abbott did not specifically detail a destination register, it is inherent that all instructions contained an indication of a destination of the result of their requested computation.

13. A device comprising:	Fig. 7a, showing a "device".
a processor	Fig. 1, showing a "processor".
that is configured to execute instructions	Col. 6, lines 7-12, detailing that the processor executes instructions.
that identify: an operation, one or more source registers, and a destination register,	See the included Hamacher et al. textbook detailing that it is inherent that all instructions identify an operation, one or more source registers, and a destination register.
each register including a word that includes a plurality of bits,	Col. 4 line 67 detailing that each register is 32 bits wide, 32 being a plurality of bits.
the processor including:	
a first connection circuit	Fig. 2, 202, showing a "REARRANGEMENT CIRCUIT" which is "a first connection circuit" as claimed.
that receives a word from a source register of the one or more source registers	Fig. 2 details that element 202 receives an "INPUT" of 48 bits, fig. 1 details that element 114 (which is shown in more detail in fig. 2) receives input from register banks 104 and/or 106 via "SELECTOR UNIT" 108.
and provides an operand word that includes a plurality of bits	Fig. 2, showing that element 202 provides a 48 bit output word, 48 bits being a plurality of bits.

that are arranged in a bit order based on a programmed set of configuration data corresponding to the operation,	Fig. 2, 202, fig. 2 shows details of the "PROGRAMMABLE LOGIC DATAPATH" of fig. 1, unit 202 of fig. 2 is a "REARRANGEMENT CIRCUIT" which reorders input data bits (col. 6 line 37 to col. 7 line 25), note specifically col. 6 lines 54-58 detailing that unit 202 allows for "selectively routing any of the 48 input bits to its output" and lines 61-63 detailing "Thus, the 48 multiplexing circuits operate to dynamically select and/or rearrange (i.e., modify relative bit positions) the input bits" which is an "ordering function". Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "REARRANGEMENT CIRCUIT" and therefore, the DCV is providing "an input ordering instruction" because it "instructs" the "REARRANGEMENT CIRUCIT" as to how to rearrange (reorder) the bits.
a configurable function circuit that provides a result word based on the operation word and the programmed set of configuration data,	Fig. 2, "SELECTIVE FIELD NEGATION CIRCUIT" 210, shown receiving 48 bits of data from "REARRANGEMENT CIRCUIT" 202. Element 210 being" being a "configurable function circuit". Box 210 is also shown providing an output 48 bits, this output 48 bits being "a result word".. Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "SELECTIVE FIELD NEGATION CIRCUIT".

| a second connection circuit that receives the result word and provides an output word for storing in the destination register that includes a plurality of bits that are arranged in a bit order based on the programmed set of configuration data. | Fig. 4, 410 "TRANSPOSITION CIRCUIT", fig. 4 is an internal view of element 400 of fig. 2, the operation of which is detailed at col. 12 line 1 to col 13 line 43. Note specifically col. 12 lines 8-10 which state "The set of multiplexers provide optional transposition (i.e., positional interchange) of rotate bit groups" where "positional interchange" is a change in the ordering of bits and is an ordering instruction.

Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "TRANSPOSITION CIRCUIT" and therefore, the DCV is providing "an output ordering instruction" because it "instructs" the "TRANSPOSITION CIRCUIT" as to how to transpose (rearrange, reorder) the bits.

Note, as detailed in the claim construction section above, applicants claims are written using inclusive or open-ended language and as such, the claims do not exclude there being additional elements present between the claimed "programmable logic blocks" and claimed "second connection circuit". The open-ended language allows for "other elements [to] be added and still form a construct within the scope of the claim." (Genentech, Inc. v. Chiron Corp.) The "other elements" being elements 402, 404, 406 and 408 which in the prior art form a conduit for the bits from the programmable logic blocks to the second connection circuit.

Note that fig. 4 details that the output from element 400 is input to optional element 214 of fig. 2, fig. 2 shows that the output of optional element 214 is the "OUTPUT" of block 114, and fig. 1 shows that the output of block 114 is carried over lines 120 to register banks 104 and 106.

Col. 5 lines 31-35 and 54-60, detailing that the DCV configures the system, which includes configuring the above referenced "TRANSPOSITION CIRCUIT". |

As to claim 14, Abbott taught an input port ("INPUT" of 48 bits on left hand side of fig. 2) that was configured to receive the word from the source register (fig. 2 "INPUT" receives data from source registers 104, 106 on fig. 1 via selector 108), and to provide the word to the first connection circuit ("INPUT" on fig. 2 is shown connecting to box 202, the "first connection circuit").

As to claims 15 and 16, Abbott taught an output port ("OUTPUT" of 64 bits on right hand side of fig. 2) that was configured to receive the output word from the second connection circuit (fig. 2 shows

"OUTPUT" receiving data from unit 212 via conduit 214, unit 212 containing the "second connection circuit"), and to provide the output word to the destination register (fig. 1 shows output from element 114 (fig. 2) connecting via line 120 to register banks 104, 106).

As to claim 17, Abbott taught that the processor (fig. 7a) included one or more other function circuits (702a ... 702, fig. 2, 214, fig. 4, 402, 404, 406, 408, 412, 414) configured to receive one or more words from the plurality of registers (input to 414 comes from register banks 104, 106 via selector 108 on fig. 1) and provide one or more other result words to the plurality of registers (output from 114 shown connecting to registers 104, 106 via line 120 on fig. 1).

As to claim 18, Abbott taught an instruction issue unit that was configured to provide the instructions to the processor (col. 1 lines 38-39, in order for an "ADD" to be "issued" an "issue" unit must be present in the CPU).

As to claims 19 and 20, Abbott taught a configuration control circuit that was configured to provide the set of configuration data corresponding to the operand based on the configuration instruction from the issue unit (fig. 1, 110, 112, fig. 2, 112).

(10) Response to Argument

At pg. 7 of the brief, appellant argues:

"Abbott fails to teach an output ordering instruction, and fails to teach routing logic output bits to a destination register specified in an instruction based on the output ordering instruction"

This argument is not persuasive because as explained in the rejections above, the element of Abbott that contains the equivalent to applicant's output ordering instruction is Abbott's DCV. This equivalence is reasonable because applicant's claims (e.g., see claim 1) recite that the "output ordering instruction" (claim 1 lines 5-6) is utilized to control the claimed "second programmable connection circuit" (claim 1 lines 18-20). In Abbott's system, Abbott defines the DCV as having the functionality of configuring and controlling all the elements of the system (col. 5 lines 30-35 and 46-60):

"Other of these sub-fields are further decoded by the decoding logic unit 112, to select other parts of a <u>direct control vector (DCV)</u> to **program/configure** various portions of the field programmable

device 100. These sub-fields are called "indirect ICV sub-fields"."

"In one embodiment, the decoding logic unit stores 16 DCVs for the subset selection portion (described later herein) of the programmable logic datapath 114 in a random access memory (RAM) and 16 DCVs in a ROM. The memory width needed to control this portion of the programmable logic datapath in this embodiment is 1536 bits, while the length of the indirect ICV sub-field that addresses this memory is 5 bits. Other memories control other portions of the selected DCV, applying them **as control signals** to the selector unit 108, the register bank 106, the register bank 104, other parts of the programmable logic datapath 114, and/or the programmable arithmetic datapath 116. In one embodiment, the total DCV length when all the fields are accounted for is 2669 bits."

Therefore, because Abbott's DCV programs, configures, and controls all aspects of the prior art device, it programs, configures, and controls Abbott's "second programmable connection circuit", and is performing the same task as that claimed by applicant for the "output ordering instruction".

At pg. 8 of the brief, appellant argues:

"Using this abbreviated interpretation of applicants' claim, the Office action fails to identify where Abbott teaches an output ordering instruction, and fails to identify where Abbott teaches routing logic output bits to a destination register specified in an instruction based on such an output ordering instructions."

This argument is not persuasive because as detailed in the rejections above and in the response to the previous argument, Abbott's DCV is the element that includes the claimed "output ordering instruction"

As to applicant's second argument it should first be noted that the scope applicant's arguments above are not commensurate with the scope of applicant's claim language. The claim language states merely:

"and selectively route the logic output bits to provide the plurality of output bits, based on the output ordering instruction." (claim 1 lines 19-20)

Applicant's claim language applies the function of "rout[ing]" only to "the output bits" for the purpose of "providing the output bits". It does not apply the routing function to routing bits to a destination register.

"The invention disclosed in Hiniker's written description may be outstanding in its field, but the name of the game is the claim." *In re Hiniker Co.*, 47 USPQ2d 1523, 1529 (Fed. Cir. 1998)

With this in mind, all Abbott's "second connection circuit" need perform, as per that which is

claimed, is "rout[ing] of the logic output bits to provide the plurality of output bits". As seen from fig. 2, logic output bits pass from box 210 to box 212, and while in box 212 as shown in fig. 4, the bits are routed (their position from input to output is modified) by box 410. Whereupon, output bits that have been routed are provided at an output of box 212 on fig. 2 and forwarded to register banks 104 and 106 via a conduit formed by optional element 214 (fig. 2), and line 120 (fig. 1). Accordingly, that which is claimed is met by the prior art.

At pg. 8 of the brief, appellant argues:

"Abbott does not teach that the output of the transposition circuit 410 is provided to a destination register that is specified in an instruction, as specifically claimed in claim 1."

This argument is not persuasive because applicant's argument is again not commensurate with the scope of applicant's claim language. Applicant's claim language, as discussed in the immediately previous argument response, only states that the claimed "second programmable connection circuit" (which is equivalent to Abbott's unit 410 in claimed function) "provide[s] the plurality of output bits" (claim 1 lines 19-20).

Applicant appears to be mixing together the claimed "unit output" (claim 1 lines 10-11) with the claimed "second programmable connection circuit" (claim 1 lines 18-20) in the arguments. However, the claimed "unit output" simply states that it is for "outputting a plurality of output bits to a destination register specified by the instruction. The claim language places no restrictions in any way upon what may or may not happen to the bits that are "provid[ed]" by the "second programmable connection circuit" before they are output by "the unit output". As was detailed in the rejection above, applicant's claims utilize open-ended claim terminology:

Genentech, Inc. v. Chiron Corp., 112 F.3d 495, 501, 42 USPQ2d 1608, 1613 (Fed. Cir. 1997) ("Comprising" is a term of art used in claim language which means that the named elements are essential, but other elements may be added and still form a construct within the scope of the claim.)

Accordingly, the claim language itself allows that "other elements may be added and still form a construct within the scope of the claim".

At pg. 8 of the brief, appellant argues:

"Block 410 does not provide the bits "that ultimately are stored in the register file" as asserted in the final Office action."

This argument is not persuasive for the same reasons as the response to the previous arguments above. Applicant's claims utilize open-ended terminology which allows that "other elements may be added and still form a construct within the scope of the claim". Box 410 provides bits, which after passing through elements 412 and 414 on fig. 4 are output to optional element 214 on fig. 2, whereupon they are output from box 114 on fig. 1 and carried via line 120 to register banks 104 and 106. Accordingly, box 410 provides bits that ultimately are stored in the register file, and the open-ended claim language is indeed met by the applied prior art.

At pg. 9 of the brief, appellant argues:

"Using the interpretation advocated by the Examiner, any and all of the elements within Abbott's data path, including the original source registers, qualify as elements that are "[a]ccordingly ... 'providing the plurality of bits' that ultimately are stored in the register file." Such an interpretation renders the claimed limitation of "selectively [routing] the logic output bits to provide the plurality of output bits [that are output to the destination register], based on the output ordering instruction" (claim 1, last two lines) meaningless"

This argument is not persuasive for the reasons presented above in response to previous arguments. Applicant's decision to utilize open-ended claim language allows that "other elements may be added and still form a construct within the scope of the claim."

Furthermore, this particular argument by applicant points out exactly how applicant is reading limitations into their claim language which do not expressly exist in the claim language itself. Applicant has annotated with square brackets in the quote of their claim language above the additional limitations that they are reading into the claim language. Those additional limitations simply do not exist in the claim language as it is currently written. While the Examiner is fully cognizant of the fact that applicant's "invention" as disclosed in the specification differs from Abbott's invention as disclosed in the prior art, in this case, as in all patent cases, "the name of the game is the claim".

"The invention disclosed in Hiniker's written description may be outstanding in its field, but the name of the game is the claim." *In re Hiniker Co.*, 47 USPQ2d 1523, 1529 (Fed. Cir. 1998)

At pg. 9 of the brief, appellant argues:

"One of skill in the art would not consider an intermediate transposition circuit that outputs bits to subsequent internal logic circuits and further processing units, as taught by Abbott, to be identical or equivalent to a connection circuit that selectively routes logic output bits to a destination register that is specified in an instruction based on an output ordering instruction as claimed by the applicants."

This argument is not persuasive because what one of skill in the art would consider is not the pertinent point. The meaning of terms in applicant's claims is a matter of claim construction. And as detailed numerous times in the previous pages, numerous court cases have decided and reinforced the claim construction meaning of open-ended claim language as "other elements may be added and still form a construct within the scope of the claim."

At pg. 10 of the brief, appellant argues:

"As noted above, Abbott does not teach programming a connection circuit to perform a routing of outputs of a programmable logic blocks to bits of a result data word that is identified in an operation that occurs in a task."

This argument is not persuasive because as discussed in great detail in the rejection and arguments response above, Abbott's DCV is utilized to program / configure / control all elements of the prior art system, which includes "programming a connection circuit" as claimed. As further discussed in the rejection, the connection circuit is for "routing" bits of a result data word where "routing" means rearrange or reorder. Accordingly, Abbott anticipates that small portion of their invention which applicant has chosen to claim.

At pg. 10 of the brief, appellant argues:

"As noted above, Abbott's element 410 is an intermediate module whose output is coupled to further logic blocks within a bank 400 of Abbott's reduction network 212. Abbott's element 410 does not output its result to a destination identified in a program instruction as specifically claimed in claim 6."

This argument is not persuasive because as discussed in great length in the rejection and arguments response above, applicant's open-ended claim language allows that "other elements may be added and still form a construct within the scope of the claim."

At pg. 11 of the brief, appellant argues:

"Specifically, and/or additionally, with regard to claim 7, upon which claims 8-12 depend, Abbott fails to teach a data processing device that includes a programmable logic block that provides an

output word of bits in a first output bit order based on received configuration data, and a second connection circuit that receives the output word and provides therefrom a word of bits in a second output bit order that is stored in a destination register based on the received configuration data."

This argument is not persuasive because as detailed in the rejection and argument responses above, Abbott provides all of these argued functions and structures.

At pg. 11 of the brief, appellant argues:

"Specifically, and/or additionally, with regard to claim 13, upon which claims 14-20 depend, Abbott fails to teach a device that includes a processor that executes instructions that identify an operation, one or more source registers, and a destination register, wherein the processor includes a configurable function circuit that provides a result word based on the operand word and a programmed set of configuration data, and a connection circuit that receives the result word and provides an output word for storing in the destination register that includes a plurality of bits that are arranged in a bit order based on the programmed set of configuration data."
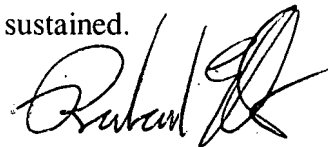
This argument is not persuasive because as detailed in the rejection and argument responses above, Abbott teaches all these argued elements and functions to the extent that they actually exist in applicant's claim language.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Richard Ellis

RICHARD L. ELLIS
PRIMARY EXAMINER

Conferees:

Lynne H. Browne
Appeal Practice Specialist, TQAS
Technology Center 2100

Eddie Chan
SPE
Art Unit 2183

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100